

Pulling Value Lean Kanban

Brian Anderson

brian@capcal.com

(214) 240-2429

History

- Waterfall
 - Heavy requirements
 - Long development cycles
- Lean Manufacturing and Agile Development
 - Eliminate waste
 - Amplify learning
 - Decide as late as possible
 - Deliver as fast as possible
 - Empower the team
 - Build integrity in
 - See the whole

•Lean Software Development – Mary Poppendieck and Tom Poppendieck

Agile Development

- The Agile Manifesto
 - We value . . .
 - Individuals and interactions over processes and tools
 - Working software over comprehensive documentation
 - Customer collaboration over contract negotiation
 - Responding to change over following a plan
- Scrum Components
 - Backlog
 - Epics, features, stories
 - Planning meetings
 - Velocity
 - Estimating / tee shirt sizing
 - Story points
 - Stand-ups
 - Burn downs
 - Owner demos
 - Retrospectives

- iterative and incremental development
- requirements and solutions evolve through collaboration
- self-organizing, cross-functional teams
- encourages rapid and flexible response to change

- Twelve principles underlie the Agile Manifesto, including:
 - Customer satisfaction by rapid delivery of useful software
 - Welcome changing requirements, even late in development
 - Working software is delivered frequently (weeks rather than months)
 - Working software is the principal measure of progress
 - Sustainable development, able to maintain a constant pace
 - Close, daily co-operation between business people and developers
 - Face-to-face conversation is the best form of communication (co-location)
 - Projects are built around motivated individuals, who should be trusted
 - Continuous attention to technical excellence and good design
 - Simplicity
 - Self-organizing teams
 - Regular adaptation to changing circumstances

•RAD RUP Spiral XP Lean Scrum
 V-Model TDD

- Problems with Scrum
 - Customers don't get story points
 - PMs spend time trying to convert points into days to predict future deliveries
 - When did you last take into account the entire value stream when estimating?
 - e.g. time to deploy, freezes, UAT unavailability
 - With real data you can make real commitments
 - Customer/Product Owner get frustrated that they have to wait for a sprint to end before new work will be considered
 - With Kanban you can dispense with sprints.
 - As soon as 1 of those items is complete they pull in the next priority item from the queue. Therefore the wait time to inject a new request is minimal
 - Planning meetings are time consuming and poorly attended

Intro to Kanban

- Kanban translated literally: “**Kan**” means visual, and “**ban**” means card or board
 - Pull customer value as quickly as possible
 - Downstream processes pull WIP only when they are ready
 - Provide a continuous flow
 - Eliminate waste
 - Focus on continuous improvement
- Excess work in progress is considered waste
 - Don’t build features that nobody needs right now
 - Don’t write more specs than you can code
 - Don’t write more code than you can test
 - Don’t test more code than you can deploy

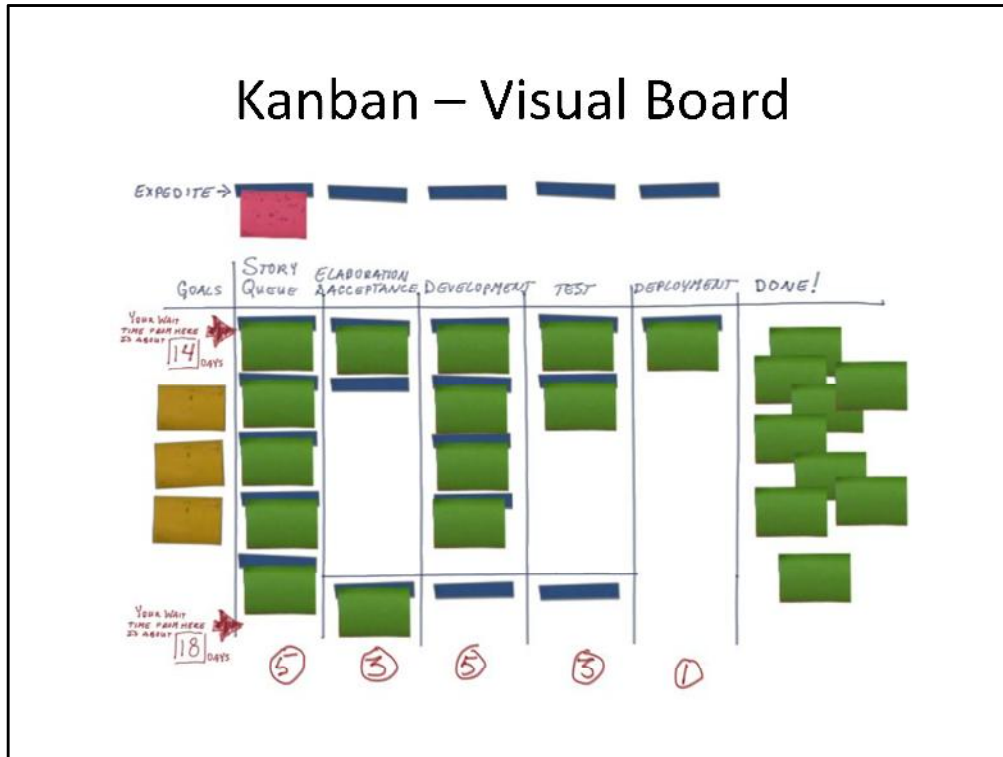


A lot of terminology in Lean software development comes from Japan and from the [Toyota Production System](#) in particular

• Prius door example

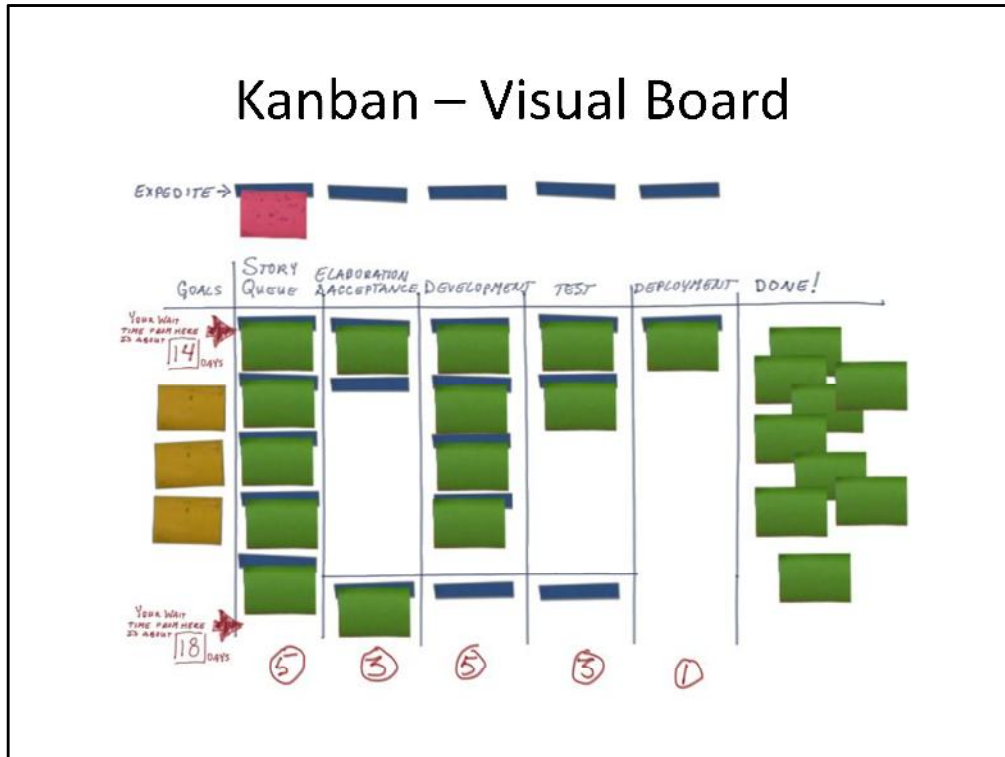
- stack of 10 or so doors
- sitting on top of the 5th door in the stack is a card — a Kanban card — that says “build 10 doors.”
- Take card to the guy who builds doors. He’s been doing other things to keep busy while waiting. The important thing here is that he’s NOT been building Prius doors.
- He takes your Kanban card and begins to build doors.
- You go back to your workstation, and just a bit before your stack of doors is gone, the door guy comes back with a stack of 10 doors.
- A Kanban card is slid in between doors 5 & 6. You got the doors just in time.
- Kanban is a pull system in which downstream processes pull WIP only when they are ready.
- It doesn’t do the Toyota factory any good to build doors faster than they can assemble cars. It just wastes money on excess doors, and parts of doors.
- Excess work in progress is considered to be waste in Lean manufacturing.
 - Mudra is Japanese for waste

Kanban – Visual Board



- Kanban starts with mapping the value stream
 - Concept to Cash
 - From the lips of the customer to the production system
 - Because this is a Kanban board we limit the amount of work in progress
 - The numbers written on the bottom limit the number of stories allowed at each station.
1. Goals — the big things we’re driving at that compel us to build software to achieve these goals.
 - columns aren’t set.
 - aren’t job titles
 - focus on the story and what it needs to mature
 - Each process step column is divided into two parts:
 - The top is used for stories currently in progress in that phase.
 - The bottom is the buffer for completed work
 - When we set limits for work in progress, we’ll set a total number for the process step that includes both “in process” and the “finished buffer” for that process step.

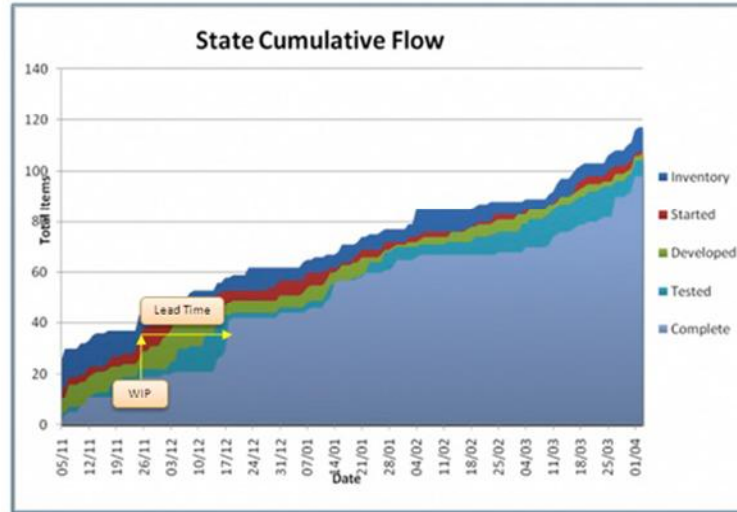
Kanban – Visual Board



- Minimal Marketable Functionality (MMF)
 - A chunk of functionality that delivers on the customer’s requirements and is capable of returning value when released.
 - Competitive differentiation
 - Revenue generation
 - Cost saving
 - Brand projection
 - Enhanced loyalty
- What Do I Work On Next?
 - Its easier to start new work than it is to finish something
 1. Can you help progress an existing Kanban? Work on that.
 2. Find a bottleneck and work to release it.
 3. Pull in work from the queue.
 4. Find other interesting work.
- Cycle time and Lead time
 - Lead time is what the customer sees.

Cause and effect

The amount of work in process, and its effect on the time to deliver that work.



Scrum and Kanban

- Scrum
 - Fixed **timeboxes**
 - Incremental cadence
 - Tasks and estimates
 - Track **velocity**

 - **Scrum Master owns process**

 - Ceremonies are required
 - **Quality is inspected in**
- Kanban
 - No Timeboxes
 - **Continual flow** of MMFs
 - No task estimates
 - Track Flow
 - Queues and WIP
 - **Cycle time**
 - **Team owns process** and is responsible for continuous improvement
 - Ceremonies are optional
 - **Quality is embedded in**

All the right conditions for continuous improvement

- Work in process is limited
- Cycle Time is managed
- Its a highly transparent and repeatable processes

Ceremonies

- Standups
 - Facilitator enumerates work, not people
 - The board shows the status, instead the focus should be on the exceptions.
 - When enumerating the work it is useful to traverse the board from right to left (downstream to upstream) in order to emphasize pull.
 - Do we have a bottleneck? (look for congestion or gaps in the queues) Do we have a blocker not dealt with? Are we keeping to our work in process limits? Are priorities clear?
 - what we did yesterday, planning today
- Retrospectives
 - More choice on when and how to reflect and improve
 - After a feature has been deployed? When we have to “Stop the line”? Weekly mini retrospective?
 - Look back at the last week [max 5 min] (What happened? Are we satisfied? Should we adjust WiP limit?)
 - Team picks one thing to improve on for upcoming week [max 2 min]
 - Write this improvement goal down on top of Kanban board

• Bugs and Rework

- There are 3 solutions:
 - Raise a bug ticket, place it in the dev ready queue, mark the item it relates to as blocked
 - Stop the line and fix the issue
 - Use the emergency queue

Summary

- Phase-based development (waterfall) transfers the entire batch from state to state
- Existing agile development transfers small batches from state to state (iterations)
- Kanban's goal is to transfer one piece of value at a time as quickly as possible

Rather than focusing on being **Agile** which may (and should) lead to being successful, Kanban focuses on becoming **successful**, which may lead to being Agile.

Start with what you do now

Modify it slightly to implement pull

Use a transparent method for viewing work, and organizing the team

Limit WIP and pull work when the team has capacity

Evolve from there by recognizing bottlenecks, waste and variability that affect performance

To start match your work in process to your current capacity.

This will be defined by whatever you have least of e.g. analysts or testers.

Take an evolutionary approach and keep changes small and incremental.